VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SYSTEMS

SLEDOVÁNÍ ODEZEV SÍŤOVÝCH SLUŽEB

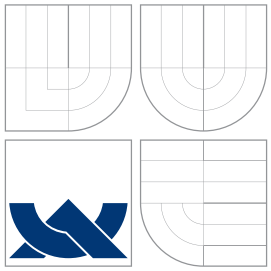
BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE PAVEL LAJCZYK

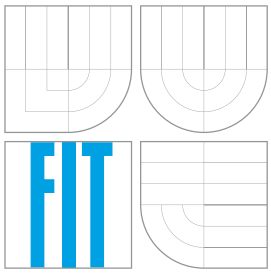
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SYSTEMS

SLEDOVÁNÍ ODEZEV SÍŤOVÝCH SLUŽEB

NETWORK AND APPLICATION RESPONSE TIME MONITORING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL LAJCZYK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ TOBOLA

SUPERVISOR

BRNO 2011

Abstrakt

Tato práce popisuje základní metody sledování odezev sítě a síťových služeb, zejména se zaměřuje na aktivní a pasivní monitorování. V další části práce jsou specifikovány požadavky na monitorovací aplikaci, která má sloužit ke sledování široké škály služeb. Na základě těchto požadavků je pak proveden návrh a implementace monitorovacího systému s webovým rozhraním, které umožňuje správu monitorování a zobrazení naměřených výsledků.

Abstract

This bachelor's thesis describes common techniques of network and network applications response time monitoring, especially it deals with active and passive monitoring. In the next chapters of this thesis there are suggested requirements for a monitoring application. Then there are provided design and implementation of such monitoring application with web interface for management of monitoring and with graphs of measured results.

Klíčová slova

Monitorování sítě, sledování odezev, aktivní monitorování, pasivní monitorování, rrdtools, démon

Keywords

Network monitoring, response time monitoring, active monitoring, passive monitoring, rrdtools, daemon

Citace

Pavel Lajczyk: Sledování odezev síťových služeb, bakalářská práce, Brno, FIT VUT v Brně, 2011

Sledování odezev síťových služeb

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Toboly

.....

Pavel Lajczyk

8. května 2011

Poděkování

Chtěl bych poděkovat především svému vedoucímu Ing. Jiřímu Tobolovi za odborné vedení při řešení této bakalářské práce.

Obsah

1 Úvod

V dnešní době, kdy čím dál větší objem komunikace probíhá prostřednictvím internetu, může i krátké přerušení spojení způsobit velké problémy. Výpadek internetové služby může ochromit akademické i jiné interní nebo veřejné sítě. Uživatelům či pracovníkům, jejichž činnost je na dané službě závislá, může podobná událost značně stížit či přímo znemožnit práci. Zákazník, kterému je služba nabízena, se při její nedostupnosti raději obrátí ke konkurenci, jež se mu může jevit jako spolehlivější. Výpadek služby tedy často vede k tomu, že ztratíme potencionálního zákazníka, případně příjdeme o důležitá data. U některých typů služeb např. v dopravě nebo ve zdravotnictví může dojít dokonce k ohrožení zdraví lidí. Každý výpadek pak zpravidla vede k finančním ztrátám, které rostou jednak s dobou nedostupnosti služby a také s frekvencí s jakou k tomuto jevu dochází.

Přesto je sledování stavu sítě mnohdy podceňováno a je vnímano spíše jako nadstandard, bez kterého se dá obejít, protože není bezprostředně nutný k fungování síťové aplikace či služby. Podstatnou část doby řešení problému výpadku služby tvoří čas, který uplyne od vzniku problému do chvíle, než se vůbec dovíme, že naše služba není dostupná nebo že nepracuje tak jak by měla. Nicméně jsou to často právě až koncoví uživatelé, kteří vzniklý problém v síti odhalí a upozorní na něho. To už je bohužel krajní situace, ke které by mělo docházet jen zřídka, nebo pokud možno vůbec. Jednak aplikace nebo služba, která očividně vykazuje chybovost, značně podkopává důvěryhodnost poskytovatele služby u jeho klienta. Zároveň je v této fázi velmi obtížné zjistit pravou příčinu problému. Koncový uživatel, který na něho poukázal, má pravděpodobně jen mlhavou představu o tom, jak je daný systém implementován a jak funguje, tudíž jeho popis vzniklých potíží nemusí být dostatečně přesný a vést tak k rychlému vyřešení kritické situace. Lokalizovat chybu může být obtížné zejména ve chvílích, kdy je nedostupnost služby způsobena souhrou více okolností, které se mohou vyskytovat náhodně. Tyto faktory značně ovlivňují rychlost, s jakou se nám podaří uvést

systém zpátky do plného provozu. Dobrý monitorovací nástroj dokáže zmírnit negativní dopady výpadku. Můžeme pomocí něj zjistit, kdy k problému došlo a co k němu vedlo. Díky monitorování máme také přehled, co se v naší síti děje, kdy je zatížení linky velké a které služby jsou nejnáročnější na přenosové pásmo. Tyto informace jsou pak užitečné pro optimalizaci síťových nastavení. Lze tak účinně a preventivně předcházet potížím.

Tématem této bakalařské práce je sledování odezev síťových služeb a aplikací. Cílem je zejména zdůraznit důležitost a užitečnost monitorování. V následující kapitole budou nejdříve představeny a porovnány nejběžnější techniky sledování sítě. Další část práce se potom bude věnovat návrhu monitorovací aplikace a její následné implementaci.

2 Sledování síťových služeb

V této kapitole budou nejdříve představeny základní prostředky pro sledování stavu sítě a síťových služeb. Dále budou předvedeny a rozebrány základní metodiky monitorování a jejich principy, také ukážeme silné a slabé jednotlivých typů monitoringu.

2.1 Základní prostředky pro monitorování

2.1.1 ICMP

ICMP neboli Internet Control Message Protocol je jedním ze základních internetových protokolů. Funguje na síťové vrstvě modelu ISO/OSI a je definován v dokumentu RFC 792 [12]. ICMP slouží k zasílání diagnostických zpráv při problémech v komunikaci IP protokolu. Tento protokol není navržen tak, aby byl absolutně spolehlivý, a proto není zaručeno, že chybová zpráva bude vždy doručena.

Nejčastější ICMP zprávy jsou Echo Request a Echo Reply. Tyto bývají využívány některými aplikacemi, nejznámější např. ping, pomocí kterého lze zjistit, zda je cílový stroj dostupný a jaká je jeho odezva. ICMP je elementárním diagnostickým prostředkem, přestože je v poslední době jeho pověst negativně ovlivněna velkým množstvím síťových útoků, ke kterým je zneužíván. Jedná se o tzv. Denial Of Service útoky (DoS), tedy o odepření služby. To může být provedeno např. zneužitím ICMP zpráv Time Exceeded a Destination Unreachable, zahlcením komunikace pomocí ping požadavků nebo dalšími metodami jako ICMP Smurf či tzv. Ping of death.

2.1.2 SNMP

Dalším protokolem, který slouží ke spravování a monitoringu sítě je Simple Network Management Protocol. Ten je specifikován ve standardu RFC 1157 [3]. Novější verze protokolu

obsahují i podporu autentizace (SNMPv2) a šifrování (SNMPv3). Pomocí tohoto protokolu je možné sledovat stav různých zařízení v síti, zpravidla routery, switche, ale i tiskárny a další prvky síťové infrastruktury za předpokladu, že podporují protokol SNMP. SNMP pracuje na principu klienta a serveru. Klientem je strana monitorovaná, také nazývaná jako agent. Server je strana monitorující, označovaná jako manager. SNMP umožňuje sledovat a nastavovat celou škálu hodnot, podle toho, jakého typu je sledované zařízení. Jednotlivé sledované hodnoty na zařízeních jsou jednoznačně identifikována pomocí OID (Object Identifier) a jsou ukládany v databázi MIB (Management Information Database).

SNMP může fungovat dvěma způsoby. Buďto aktivně, a to tak, že správce zasílá agentovi dotazy a ten na ně okamžitě odpovídá. Nebo SNMP pracuje v pasivním režimu za využití tzv. SNMP traps, kdy agent odešle správci zprávu o svém stavu jen v případě, že nastane nějaká předem definovaná událost (např. teplota procesoru překročí určitou hranici).

Nevýhodou tohoto protokolu je, že neumožnuje managerovi získat větší souhrn dat nebo jejich analýzu jen pomocí jediného dotazu. Řešením je pravidelné a časté dotazování, tzv. polling, ze strany serveru směrem k agentovi. Tento přístup ale nese riziko zvýšeného zatížení přenosového pásma.

2.1.3 NetFlow

Protokol NetFlow byl navržen společností Cisco jako prostředek pro monitorování IP toků v síťovém provozu. IP tok je posloupnost paketů, která má shodnou zdrojovou a cílovou adresu, port a druh protokolu. Pro každý tok jsou sbírany různé údaje jako např. délka trvání toku, počet přenesených paketů a další.

Klasická NetFlow architektura zpravidla obsahuje NetFlow kolektor a směrovače, jež jsou zároveň NetFlow exportéry. NetFlow exportér monitoruje provoz na lince, ke které je připojen. Z naměřených dat vytváří statistiky, které následně exportuje na NetFlow kolektor. Kolektor je datové úložiště, kde dochází k ukládání statistik do databáze. Tyto statistiky obvykle pocházejí z více NetFlow exportérů. Nevýhodou klasické architektury je, že snižuje výkon směrovačů, což je řešeno tím, že se při tvorbě statistik bere v úvahu jen každý N-tý paket. To má za následek snížení přesnosti statistik.

Výše zmíněné problémy jsou v moderních architekturách eliminovány nahrazením NetFlow exportérů pasivními NetFlow sondami. Nasbíraná data odesílají na kolektor přes dedikované spojení a pro ostatní zařízení na lince jsou transparentní, což je kladem z hlediska

bezpečnosti. Dalším plusem jsou nižší finanční náklady v porovnání s tradičními NetFlow architekturami.

2.2 Aktivní a pasivní monitorování

Při studování problematiky sledování síťového provozu lze rozlišit především dva základní přístupy. Jedná se jednak o aktivní monitorování, druhým způsobem je potom sledování pasivní. Filozofie obou metod je, dá se říct, protikladná. Přesto se hojně využívají obě a zejména jejich propojením je možno vytvořit silný a kvalitní monitorovací systém.

2.2.1 Aktivní monitorování

Základní myšlenka aktivního sledování odezev spočívá v co nejvěrnějším napodobení reálného provozu na síti. Jedná se vlastně o uměle vytvořenou simulaci chování uživatele služby, proto je tento druh monitorování někdy také označován jako syntetický monitoring (synthetic monitoring). Aplikace postavená na tomto principu aktivně vstupuje do dění na síti, a to tak, že cílovému prvku síťové infrastruktury posílá data monitorovaného protokolu nebo služby. Takovýmto postupem lze jednoduše zjistit jaká je doba odezvy mezi odesláním dotazu a přijetím očekávané odpovědi. V případě, že nedostaneme odpověď žádnou, okamžitě víme o nefunkčnosti sledované služby a ihned můžeme začít vzniklou situaci řešit. Hodnoty odezev také napoví jestli komunikace probíhá v pořádku nebo jestli hrozí nebezpečí výpadku např. z důvodu zahlcení linky nebo kvůli velké ztrátovosti datových paketů. Preventivně tak můžeme předcházet možným problémům ještě předtím, než se s nimi setká skutečný uživatel, což je jednoznačně výrazným kladem tohoto typu monitorování.

Na druhou stranu aktivním odesíláním testovacích dat zatěžujeme síť dalším provozem, což nemusí být vždy žádoucí. V případě intenzivních testů, jejichž cílem je zjistit propustnost sítě, ale i při méně náročných testech, může dojít k negativnímu ovlivnění provozu skutečných uživatelů. Proto je důležité správně zvolit frekvenci, s jakou bude monitorovací nástroj provádět měření odezev syntetickým způsobem, a to tak, aby pro nás takto získané informace stále měly ještě požadovanou vypovídající hodnotu, ale zároveň, aby nedocházelo ke zbytečnému zahlcování sítě. Dalším úskalím je náročnost vytvoření věrného simulovaného provozu, protože je takřka nemožné odhadnout všechny možné situace, které mohou při reálném běhu služby nastat.

Aktivní monitoring lze implementovat na různých úrovních složitosti. Nejjednodušším přístupem je pouhé testování dostupnosti stroje a portu, na kterém služba běží. Zjistit, zda cílový počítač běží, a jaká je jeho odezva, se dá využitím ICMP zpráv Echo Request a Echo Reply. Tato funkcionalita je implemetována v programu ping. Dostupnost služby se dá ověřit pokusem o připojení na port, na kterém by měla služba podle očekávání běžet. Pokud je port otevřený, tak to znamená, že na něm služba naslouchá a tedy pravděpodobně funguje. V případě, že se port jeví jako filtrovaný nebo zavřený, tak se nejspíše jedná o chybu služby. Tato technika skenování portu nám ale bohužel neřekne, jestli služba opravdu pracuje tak, jak má. Zároveň takto nezjistíme, jaké jsou odezvy při obsluhování reálných požadavků. Ověřování dostupnosti portu lze provádět aplikacemi jako např. nmap, které nabízejí několik druhů skenování portů.

Mnohem složitější než pouhé zkoumání, zda je daný síťový prvek dostupný, je kompletní napodobení interakce uživatele s poskytovanou službou. Takto lze vytvořit velmi sofistikovaná řešení, kdy testujeme jednotlivé funkce služby, které jsou pro nás nejdůležitější a snažíme se ověřit, jestli pracují podle předpokladů a také, jak dlouho trvají jednotlivé úkony. Můžeme tak např. podrobně zkoumat jaké odezvy mají různé příkazy nad databázovou tabulkou, nebo ověřit funkčnost FTP serveru. Návrhu a implementeci podobné komplexní aplikace založené na aktivním monitoringu se budou věnovat kapitoly 3 a 4.

2.2.2 Pasivní monitorování

Pasivní monitoring, jak už je patrné z názvu, nezasahuje do síťového provozu aktivně, nýbrž pouze zachytává na síti komunikaci, která je vytvořena skutečnými uživateli. Z tohoto důvodu je tato technika občas nazývána jako Real User Monitoring (RUM).

Sběr monitorovaných dat může být uskutečněn pomocí softwarových aplikací tzv. snifferů jako např. Wireshark nebo tcpdump. Ty čtou data přímo ze síťové karty (NIC), která v případě, že je nastavená v promiskuitním režimu, nezahazuje rámce, které nejsou určeny pro její MAC adresu a umožňuje tak odchytávat provoz v síti. Další možností jsou speciální karty NMIC (Network Monitoring Interface Card), které jsou navrženy k tichému naslouchání síťovému provozu. Většinou nemají vlastní MAC adresy, takže jsou neviditelné pro ostatní síťové prvky, které jsou zapojeny do stejné sítě. Tyto karty bývají, kromě monitorování a analýzy sítě, využívány také v IDS (Intrusion Detection) a IPS (Intrusion

Prevention) systémech, což jsou systémy, které se snaží detekovat pokusy o útok (IDS), případně na útok i určitým způsobem zareagovat (IPS).

Při sledování větších sítí se často využívá i hardwarových řešení v kombinaci se softwarem. Důvodem je velký objem dat přenášených po síti v reálném čase, které je nutné zachytávát a analyzovat. Proto je nezbytné rozdělit práci mezi hardware a software, kdy hardware může provádět např. klasifikaci paketů podle specifikovaných kritérií. Díky tomuto filtrování pak softwarová aplikace už dále zpracovává mnohem menší objem dat a provádí nad nimi další operace jako např. vytvoření statistik o provozu. Hardwarová část se obvykle řeší připojením speciálního monitorovacího adaptéru do sítě, buďto pomocí rozbočovače tzv. splitteru nebo připojením k portu na přepínači (switchi), který je nastaven na kopírování paketů ze síťového provozu (tzv. port mirroring).

Pasivní monitorování je výhodné při zjišťování výkonu sítě, a určení náročnosti jednotlivých služeb a umožnují odhalit např. to, které aplikace spotřebovávají nejvíce přenosového pásma. Takto nabyté informace je potom možno využít při optimalizaci sítě. Výhodou je, že naměřená data jsou založená na skutečném provozu. Z toho ale plyne i nevýhoda, a to ta, že případný problém je odhalen až po jeho vzniku a tedy ve chvíli, kdy už se s ním setkal koncový uživatel.

3 Návrh

V této kapitole budou nejdříve specifikovány požadavky na monitorovací nástroj založený na aktivním monitorování. Po analýze těchto požadavků bude následně popsán návrh takového systému.

3.1 Specifikace požadavků

Ze zadání práce vyplývá, že má být vytvořena aplikace, která bude sloužit ke sledování odezev síťových prvků, jejich služeb a aplikací na nich běžících. Aplikace by měla primárně fungovat pod operačním systémem Linux. Vzniklý monitorovací systém by měl umožňovat sledování širokého spektra služeb a to tak, aby bylo co nejlépe napodobeno chování skutečného uživatele sledované služby.

Uživatel aplikace by měl mít k dispozici přehledné a intuitivní webové rozhraní. Důvodem pro vytvoření webového rozhraní namísto běžné aplikace, je zejména přítomnost internetového prohlížeče téměř ve všech v současnosti využívaných operačních systémech a tím pádem i to, že odpadá nutnost instalovat do počítače jakýkoli nový software. Pomocí webového uživatelského rozhraní by mělo být možno jednoduše přidávat a nastavovat jednotlivá sledování. Také bude sloužit k prezentaci naměřených výsledků a to nejlépe v podobě grafů, které jsou pro člověka lépe čitelné a umožňují snadnější zhodnocení zjištěných odezev a výpadků služby, než hodnoty vypsané pouze číselně v tabulkách.

Ideální by také bylo, kdyby byla výsledná aplikace lehce rozšiřitelná o další moduly umožňující monitorovaní i jiných služeb než jen těch, jejichž moduly budou v systému implicitně napevno zabudovány.

3.2 Analýza a shrnutí požadavků

Po analýze požadavků je zřejmé, co všechno bude potřeba k vytvoření požadovaného systému. Vzhledem k tomu, že má aplikace simulovat chování opravdových uživatelů, tak bude zapotřebí vybrat vhodný nástroj, který umožní napodobit fungování jednotlivých služeb. K tomuto se výborně hodí skriptovací jazyk Perl, který obsahuje velké množství knihoven pro práci se sítí. Díky těmto knihovnám nebude třeba od základu implementovat funkcionalitu jednotlivých služeb. Dalším přínosem volby tohoto skriptovacího jazyka je, že bývá často využíván k psaní různých monitorovacích skriptů a tudíž už zpravidla bývá nainstalován na většině serverových stanic a to i včetně některých knihoven potřebných k našemu účelu.

Pro provoz systému bude také nutný webový server běžící na operačním systému Linux. Na vytvoření samotného webového rozhraní bude využit značkovací jazyk HTML spolu se skriptovacím jazykem PHP, což je v současnosti asi nejrozšířenější způsob tvorby webových stránek.

Naměřené hodnoty budou ukládány do Round Robin databází za pomocí aplikace RRDTool. Výsledky budou uživateli prezentovány v podobě grafů, které budou vykreslovány také prostřednictvím RRDTool.

Vzhledem k tomu, že má systém umožnovat ukládání nastavení systému, uživatelů a jednotlivých měření, tak je nezbytné zvolit vhodný způsob k uchovávaní těchto dat. K tomuto účelu poslouží SQL databáze.

3.3 Návrh monitorovacího systému

Nyní bude popsán návrh zadaného monitorovacího nástroje. Aplikace je rozdělena na několik logických částí, které budou v této podkapitole představeny. Rozlišit lze zejména serverovou část a část klientskou. Serverová část se skládá z několika menších modulů. Tyto moduly se starají o samotná měření jednotlivých služeb, ale také o vytváření RRD databází a vykreslování grafů. Klientská část celé aplikace je reprezentována webovým rozhraním, pomocí něhož uživatel přidává a nastavuje jednotlivá sledování. Další důležitou součástí aplikace je SQL databáze, ve které jsou uložená nastavení. K ní přistupují jak klient, tak server.

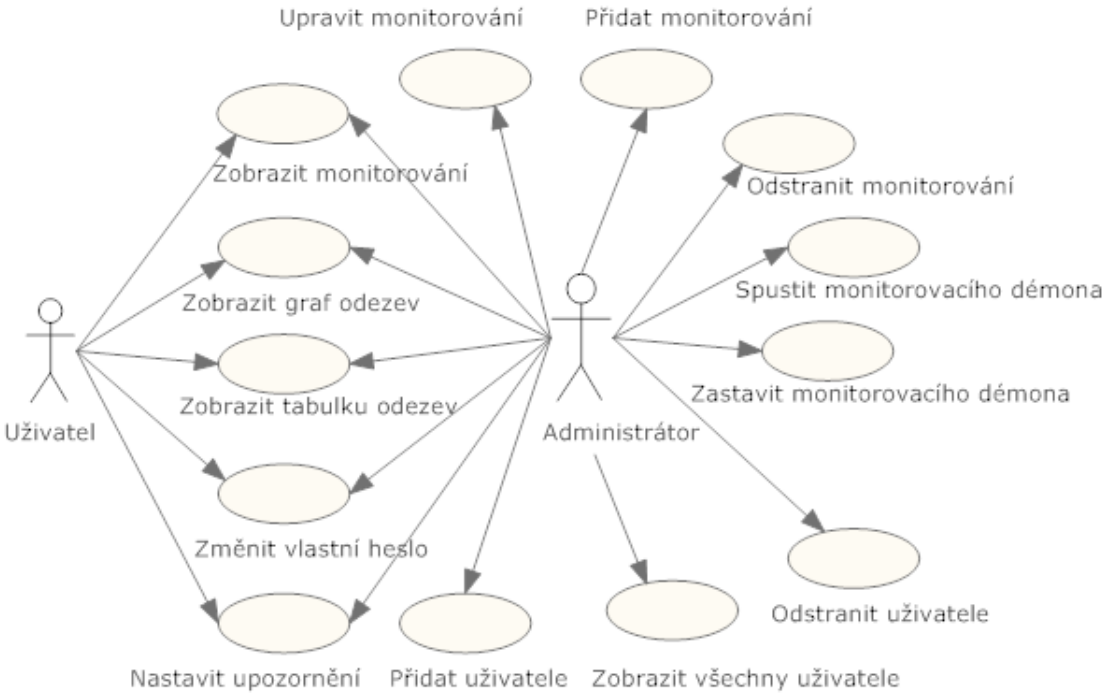
3.3.1 Monitorovací démon – server

Nejdůležitější částí celého monitorovacího systému je jeho serverová část. Protože sledování probíhá dlouhodobě, tak je nutné zajistit, aby aplikace nebyla závislá na právě přihlášeném uživateli. Toto bude provedeno tak, že serverová část bude fungovat jako démon, tedy skript běžící na pozadí systému. Tento skript potom bude spouštět nebo zastavovat samostatné monitorovací moduly. To může provádět buďto na základě nastavení, která jsou uložena v SQL databázi, nebo může přijímat konkrétní příkazy od klientské části, tedy z webového rozhraní.

3.3.2 Webové rozhraní – klient

Webová část aplikace je navrhnuta takovým způsobem, aby byla pokud možno co nejvíce intuitivní a aby uživateli umožnovala snadnou a rychlou práci. Celé rozhraní je proto zaměřeno hlavně na praktičnost a naopak se neklade takový důraz na grafické zpracování. Pomocí webu tak půjde jednoduše spouštět nebo zastavovat monitorovacího démona a provádět jeho základní nastavení. Hlavním cílem je snadné přidávání nových měření, přehledné zobrazení jednotlivých monitorování, která už jsou v systému zadána a také vhodná prezentace naměřených výsledků.

Při tvorbě webového rozhraní je důležité si ujasnit jaké role budou mít jednotliví koncoví uživatelé. Budeme předpokládat, že rozhraní budou využívat dva typy uživatelů. Prvním typem je běžný uživatel, který bude mít možnost procházet naměřené hodnoty a provádět některá základní nastavení svého účtu. Další uživatelská role bude administrátor, jež bude mít k dispozici širší škálu funkcí a větší pravomoce. Především pak možnost přidávat do systému nová monitorování, ale také bude moci zaregistrovat a odstraňovat další uživatele. Jaké mají tyto dva typy účtů práva je zobrazeno v Use Case diagramu na obr. 3.1



Obrázek 3.1: Use Case Diagram navrženého systému

3.3.3 Databáze SQL

Pro uchovávání dat o uživatelích a nastavení systému bude využit otevřený databázový systém PostgreSQL. Před vytvořením databáze je třeba vědět jaká data budeme potřebovat uchovávat a stejně tak jaký bude formát těchto dat. Databáze SQL byla zvolena z toho důvodu, že data uložená v tabulkách budou mít různorodé datové typy. Samotná struktura databáze se potom bude skládat ze čtyř tabulek.

V první tabulce budeme mít informace o všech monitorovaných adresách. Pro účely sledování budeme muset u každé adresy znát několik údajů. Především si musíme uložit název služby, kterou budeme sledovat, dále pak port na kterém služba běží. Stejně tak potřebujeme znát přihlašovací jméno a heslo u služeb, které vyžadují autentizaci. U monitorování databází je ještě zapotřebí název sledované databáze. Pro uživatele může být také užitečné datum, kdy bylo sledování do systému zadáno, proto ho také vložíme do databáze. Dále bude v této tabulce u každého sledování uchována ještě informace o tom, zda je dané monitorování aktivováno, deaktivováno nebo jestli má být sledování z databáze odstraněno.

Druhá tabulka bude obsahovat záznamy o uživatelích, kteří budou pracovat se systémem. U nich je nutné vědět především prihlašovací jméno a také heslo. Na základě těchto údajů se budou uživatelé přihlašovat do webového rozhraní. Dále si systém uchová i jejich skutečné jméno a příjmení, aby bylo administrátorům jasné o koho jde v případě řešení některých problémů se systémem. Dalším údajem je, jestli se jedná o běžného uživatele nebo o uživatele s administrátorskými pravomocemi. Poslední informace o uživateli uložená v databázi je jeho e-mailová adresa. Tu uživatel využije ke dvěma účelům. Jednak mu na ni příjde jeho přihlašovací heslo po zaregistrování do systému. Dále si pak může nastavit, že na zvolenou e-mailovou adresu bude dostávat upozornění v případě výpadku některé monitorované služby.

Ve třetí tabulce budou informace o upozorněních na výpadky služby. Budou zde uchovány e–maily uživatelů, na které se mají upozornění posílat. Také se bude ukládat hodnota odezvy, po jejíž překročení dojde k odeslání upozornění. A samozřejmě je také nutné vědet, kterého monitorování se upozornění týká.

Čtvrtá tabulka obsahuje informace o monitorovacím démonovi, respektive to, na kterém serveru běží a na jakém portu naslouchá. Tyto data poté klient využije při komunikaci s démonem.

3.3.4 Nástroj RRDTool

RRDTool [11] je open-source systém, který slouží k uchovávání různých naměřených údajů závislých na čase, dále podporuje zpracovávání těchto dat a hlavně umožňuje tyto data jednoduše vykreslovat do přehledných grafů.

Celý princip tohoto nástroje je postaven na tzv. Round Robin databázích. Výhoda tohoto druhu databází spočívá v tom, že jakmile je daná databáze jednou vytvořena, tak už má konstantní velikost po celou dobu své existence. Nehrozí tak nekontrolovatelný růst zabíraného místa, který může nastat u jiných typů databází. Této vlastnosti je docíleno tím, že jsou hodnoty do databáze ukládany cyklicky. To znamená, že jakmile je při ukládání dat dosaženo konce databáze, tak se začnou přepisovat data na počátku. Tento rys je vhodný pro ukládání velkého množství číselných údajů, které získáváme při monitorovaní síťě, proto byla zvolena právě tato aplikace. Data jsou do databáze plněna v pravidelných intervalech a pokud není v daném intervalu uložena žádná hodnota (např. při výpadku služby), tak se zaznamená tzv. „unknown“ hodnota.

RRDTool zároveň umí vytvářet grafy podle naměřených hodnot ze zvoleného časového období. Při vykreslování grafů za delší časový úsek dochází ke snížení rozlišovací schopnosti. U tohoto většího časového rozsahu lze pak určit, jestli se mají vykreslovat průměrné, minimální nebo maximální hodnoty z předchozích kratších časových úseků.

4 Implementace

Tato kapitola bude popisovat jaké konkrétní postupy byly zvoleny při tvorbě zadaného monitorovacího systému a jakým zbůsobem byly implementovány jeho dílčí součásti. Nejdříve se podíváme na základní část, tedy monitorovacího démona a další skripty na nichž celé monitorování stojí. Následně bude podrobně rozebráno webové rozhraní.

4.1 Monitorovací démon

Hlavní skript, jež je serverovou částí aplikace a který je zároveň v podstatě klíčovým prvkem celého monitorování, je implementován ve skriptovacím jazyce Perl [1]. Tento skript pracuje jako démon, to znamená, že potichu běží na pozadí systému a běžnému uživateli není jeho běh na první pohled nijak patrný.

4.1.1 Programování démona

Pro správné fungování démona [5] je potřeba při jeho programování zajistit několik věcí, bez nichž může docházet k neočekávanému chování nebo dokonce k haváriím programu. Samotné přesunutí běhu skriptu do pozadí je provedeno pomocí funkce `become_daemon` a je při tom využíváno několika POSIXových funkcí.

Nejdříve si funkcí `getppid`, která vrací číslo rodičovského procesu, ověříme jestli náhodou už skript na pozadí neběží. Pokud by funkce `getppid` vrátila hodnotu 1, tak to znamená, že je rodičem proces Init a náš skript už běží na pozadí. Tím pádem dojde k úspěšnému ukončení funkce `become_daemon`, která už nemusí provádět další kroky. Ovšem pokud funkce `getppid` vrátí jinou hodnotu než číslo procesu Init, tak to znamená, že je třeba pokračovat v přesunu běhu na pozadí.

Prvním krokem je volání funkce `fork`. Po zavolání této funkce vznikne nový proces, přičemž rodičovský proces vzápětí ukončím. Tím dojde k odpojení procesu od terminálu,

ve kterém byl spuštěn. Zároveň je také odstraněn ze skupiny procesů, ve které byl skript spuštěn. Tím pádem nehrozí, že je skript vedoucím procesem skupiny, což je nutnou podmínkou pro volání funkce `setsid`.

`Setsid` je funkce, která umožní procesu získat nové sezení (session) a proces se zároveň stane vedoucím této nové skupiny. Tím je zaručeno, že démon nebude přijímat signály od kontrolního terminálu, ve kterém byl spuštěn. Také to znamená, že zůstane běžet i po odhlášení uživatele, který ho spustil.

Dále je nutné uzavřít všechny nepotřebné souborové deskriptory (tzv. file descriptors), jelikož by potom byly děděny do všech synovských procesů, tedy jednotlivých monitorovacích skriptů, které bude démon spouštět. To by mohlo mít za následek zbytečné zabírání zdrojů operačního systému.

Vhodné je také přesměrovat standardní výstup a standardní chybový výstup do logovacího souboru. V případě chyby při běhu démona, tak budeme mít zaznamenáno proč k ní došlo.

Poslední věcí, kterou je potřeba provést, je vytvoření speciálního souboru, který si následně démon zamkne pomocí funkce `flock`. Takto je zabezpečeno, že poběží jen jedna instance démona. Pokud by byla spuštěna další instance, tak se pokusí uzamknout stejný soubor, což selže v případě, že už si ho uzamkla jiná instance.

Vzhledem k tomu, že démon bude spouštět více dalších skriptů, tak je v něm implementováno zachytávání signálů od synovských procesů, aby nedocházelo ke vzniku tzv. zombie procesů.

4.1.2 Programování serveru

Výše byl popsán způsob jakým je provedeno přesunutí běhu programu na pozadí operačního systému. Jakmile je tento základ hotov, tak můžeme začít implementovat samotnou funkcionalitu serveru. Server je napsán s využitím tzv. unixových socketů [14]. Nejdříve je funkcí `socket` vytvořen nový socket a jeho parametry jsou nastaveny funkcí `setsockopt`. Následně přiřadíme socketu port využitím funkce `bind` a poté je funkcí `listen` spuštěno naslouchání na tomto portu. Port, na kterém monitorovací démon poslouchá je implicitně nastaven na hodnotu 10001. Tuto hodnotu má ale uživatel možnost jednoduše změnit prostřednictvím webového rozhraní, v případě, že by byl výchozí port již zabrán jinou aplikací.

Běh serveru je reprezentován nekonečnou smyčkou, která běží po celou dobu fungování serveru. V tomto nekonečném cyklu dochází k přijímání příchozích spojení funkcí `accept`. Jakmile je spojení od klienta přijato, tak začne probíhat komunikace na základě velmi jednoduchého protokolu. Tento jednoduchý protokol je tvořen krátkými textovými řetězci, jež vyjadřují jednotlivé příkazy. Server může od klienta příjmout příkazy STATUS, RESTART, STOP nebo SEND DATA.

Příkazem STATUS se klient dotazuje, zda monitorovací démon běží nebo ne. Pokud server odpoví zprávou STATUS OK, tak klient ví, že je démon spuštěn. Pokud klient nedostane žádnou odpověď, tak předpokládá, že je server zastaven.

Na příkaz RESTART zareaguje server tím, že voláním funkce `stop_children` zastaví všechny spuštěné skripty provádějící monitorování nebo vykreslování grafů a poté je znovu spustí pomocí funkce `start_monitoring`. Zpráva STOP způsobí zastavení monitorovacího démona, stejně jako zastavení všech ostatních synovských skriptů.

Nejzajímavější je příkaz SEND DATA, který je následován názvem souboru. Takto si klient vyžádá soubor, který potřebuje. Může se jednat buďto o nějaký konkrétní graf, nebo databázový soubor s naměřenými výsledky monitorování.

Velmi důležitá funkce, která je implementovaná na serverové straně je funkce `start_monitoring`. Jak její název napovídá, tak slouží ke spuštění samotného monitorování. Tato funkce se nejdříve připojí k SQL databázi a načte z ní seznam adres, které se mají monitorovat. Poté pro každé toto sledování vytvoří nový proces funkcí `fork`, ve kterém se spustí konkrétní skript, který přísluší monitorování zvolené služby na dané adrese. Opakem je funkce `stop_children`, která systémovým voláním `kill` zastaví všechny synovské procesy.

4.1.3 Modul pro vykreslování grafů

Skript, který se stará o vytváření grafů, je rovněž napsán v Perlu. Monitorovací démon ho spustí při každém volání funkce `start_monitoring`, stejně jako ostatní monitorovací moduly. Tento modul si nejdříve načte z SQL databáze potřebné informace o všech aktivních monitorováních. Poté dochází v nekonečné smyčce v pravidelných intervalech k vykreslování všech grafů. Samotné vykreslování je provedeno RRD příkazem `graph`, k jehož použití je zapotřebí mít nainstalovanou perlovskou knihovnu pro práci s RRDTools. Příkazu graph je předáno několik parametrů, které určují to, jak bude graf vypadat a jaké má mít rozměry.

Nejdůležitějšími parametry ale jsou název RRD databázového souboru, z něhož se mají čerpat naměřená data pro vykreslení a také název výsledného souboru s grafem.

Pravidla pro vytváření názvu tohoto souboru jsou pevně dána. Každý obrázek s grafem je uložen ve formátu PNG. Název obrázku dále obsahuje název služby, které se monitorování týká, a také adresy. Za adresou se ještě vyskytuje číslo, jež označuje jak dlouhý časový interval graf zobrazuje. Název, který by měl graf naměřených odezev za poslední hodinu by např. při sledování služby FTP na adrese eva.fit.vutbr.cz vypadal takto: ftp–pic–eva.fit.vutbr.cz–60.png

4.1.4 Monitorovací moduly

Každý monitorovací modul, je skript napsaný ve skriptovacím jazyce Perl. Tyto skripty mají pevně daný název, který se skládá ze zkratky názvu příslušné služby a druhá část názvu je stejná pro všechny skripty, např. skript, který slouží k monitorování služby FTP má název ftp–mon.pl. Výhodou tohoto přístupu je, že tak lze snadno přidávat do systému nové moduly pro služby, které ještě systém neobsahuje.

Jednotlivé monitorovací moduly jsou spuštěny z monitorovacího démona, na základě informací, jež jsou uloženy v SQL databázi. Pro každý skript je vytvořen funkcí `fork` nový proces. V případě, že v některém modulu dojde k nějaké chybě, není tím ohrožen běh ostatních modulů a celé aplikace.

Na začátku každého modulu se nejdříve prověří, jestli existuje příslušná RRD databáze pro dané monitorování, pokud ne tak dojde k vytvoření nového databázového souboru. To je provedeno voláním RRD příkazu `create`. Příkaz `create` je proveden s patřičnými parametry jako je název databázového souboru, interval s jakým bude monitorování probíhat a maximální doba po jakou budou data v databázi uložena. Přičemž tyto poslední dva parametry značně ovlivňují velikost databázového souboru, protože určují kolik dat v databázi vlastně bude. Čím kratší je interval monitorování, tím větší bude výsledná velikost souboru.

RRD databázový soubor je identifikován podle koncovky „.rrd“. Dále se v jeho názvu vyskytuje zkratka dané monitorované služby a také adresa, na které je odezva této služba měřena. Pokud znovu použijeme za příklad službu FTP, která by běžela na adrese eva.fit.vutbr.cz, tak celý název RRD databázového souboru pro toto monitorování by byl ftp–db–eva.fit.vutbr.cz.rrd. Implementovaná aplikace ukládá naměřené hodnoty pro každé sledování až po dobu jednoho roku.

Celá aplikace má být schopna monitorovat, co největší počet služeb. Proto systém obsahuje moduly pro nejběžněji používané služby, jedná se o HTTP, ICMP, DNS, SSH, FTP, POP3, SMTP, IMAPs, MySQL, PostgreSQL. Také je možné testovat dostupnost zvoleného TCP portu na dané adrese. Implementace monitorovacích modulů pro jednotlivé služby, a stručný popis monitorovaných služeb bude probrán následně.

4.1.5 HTTP modul

Jedním z nejpoužívanějších protokolů v internetu je právě protokol HTTP, neboli HyperText Transfer Protocol. Verze HTTP/1.1 tohoto protokolu je definována v dokumentu RFC 2616 [6]. Tento protokol obvykle pracuje na portu 80 a slouží zejména k přenesou hypertextovýh dokumentů ale i dalších souborů jako jsou třeba obrázky.

Funguje na principu dotaz/odpověď, kdy klient zasílá serveru požadavek na daný soubor a server na tento dotaz odpovídá. Existuje několik druhů těchto dotazů, mezi nejvyužívanější patří určitě příkazy GET, POST a HEAD. GET je příkaz, kterým klient žádá nějaký soubor ze serveru. Naopak metoda POST slouží k zaslání dat na server, v praxi se takto většinou odesílají např. hodnoty vyplněného formuláře. Příkaz HEAD funguje velmi podobně jako metoda GET s tím rozdílem, že server v odpovědi neodesílá požadovaný soubor, ale pouze základní informace o tomto souboru, tzv. metadata. Tato metoda většinou slouží ke zjištění zda se daný soubor na serveru nachází či ne.

Název skriptu, ve kterém je modul implementován, je http–mon.pl. Modul si nejdříve načte z SQL databáze důlezité informace jako je adresa, port na kterém HTTP služba běží, ale také interval s jakým má monitorování probíhat. Pokud ještě neexistuje příslušný databázový soubor, tak dojde k jeho vytvoření. Samotné monitorování je uskutečněno v nekonečné smyčce. Zadaný interval monitorování a tím pádem i pravidelného plnění RRD databáze dodrženo uspáním skriptu využitím funkce `sleep`, která má jako parametr hodnotu intervalu načtenou z databáze. K monitorování je využito funkcí z perlovské knihovny pro práci s protokolem HTTP. Prostřednictvím funkce `write_request` z této knihovny je na zadanou adresu a port poslána žádost GET o stažení dané stránky. Pokud se stažení nepodaří, tak se jedná o výpadek. Před posláním zprávy GET a po úspěšném stažení stránky se do dvou speciálních proměnných uloží čas začátku a čas konce měření využitím funkce `clock_gettime`. Odečtením těchto dvou proměnných získáme odezvu, která je následně uložena do RRD databáze pomocí RRD příkazu `update`.

4.1.6 ICMP modul

Tento modul ověřuje dostupnost cílového stroje pomocí protokolu ICMP. Skript icmp–mon.pl načte z databáze informace o monitorované službě, adresu a frekvenci s jakou má monitorování probíhat. V případě tohoto modulu není třeba znát port, protože každá ICMP zpráva je zapouzdřena přímo v IP datagramu. Modul dále vytvoří RRD databázi pro dané sledování, pokud chybí. Následuje nekononečný cyklus, ve kterém probíhá monitoring. Při něm je využit systémový program Ping. Pingem je odeslán jeden paket na cílovou adresu. Z přijaté odpovědi je potom pomocí unixových nástrojů awk, grepu a perlovské funkce split vyextrahována naměřená odezva.

Tento odlišný postup měření odezvy oproti ostatním modulům byl zvolen z toho důvodu, že odpověď programu Ping už obsahuje časovou značku odezvy, a je tedy zbytečné ji měřit znovu jiným způsobem, který by byl navíc pravděpodobně zatížen i nepřesností, protože by byl měřen i čas spojený s režií volání a běhu programu ping. Získaná odezva se uloží do RRD databáze přislušící danému monitorování a funkce `sleep` proces uspí do doby následujícího měření.

4.1.7 DNS modul

DNS je jedním z nejpoužívanějších protokolů, přestože o něm řada uživatelů internetu ani netuší. Celý název tohoto protokolu je Domain Name System a jeho definice je popsána ve standardech RFC 1034 [8] a RFC 1035 [9]. Díky tomuto protokolu je umožněn překlad doménových jmen na číselné IP adresy a zpět. DNS protokol standardně pracuje na TCP i UDP portu 53.

Celý DNS systém je postaven na základě hierarchicky organizovaných DNS serverů, které vlastně tvoří decentralizovanou databází doménových jmen a IP adres. Existují tři typy DNS serverů a to primární DNS server, sekundární DNS server a tzv. Caching Only DNS server. Jednotlivé typy se liší podle toho zda je DNS odpověd od daného serveru věrohodná, tedy autoritativní (primární a sekundární server) nebo autoritativní není (Caching Only Server).

Modul dns–mon.pl je implementován následovně. Stejně jako u ostatních modulů i zde se nejdříve načtou data z SQL databáze, v tomto případě se jedná o adresu DNS serveru, který má provádět překlad, dále port na kterém služba běží a interval monitorování. V mo-

nitorovacím cyklu se provádí překlad adresy www.google.com na zadaném DNS serveru. Je k tomu využito knihovny Perlu `Net::DNS`, jejíž metoda `search` provádí samotný překlad adresy. Před zahájením pokusu o překlad je opět pomocí funkce `clock_gettime` uložena počáteční časová značka a po úspěšném přeložení adresy je uložen koncový čas. Získaná odezva je pak zapsána do odpovídající RRD databáze. Na další měřící cyklus se čeká pomocí funkce `sleep`.

4.1.8 FTP modul

FTP neboli File Transfer Protocol je platformě nezávislý protokol pracující na aplikační vrstvě síťového modelu ISO/OSI a funguje na principu klient/server. Definice FTP protokolu je uvedena v dokumentu RFC 959 [13]. FTP slouží k přenášení dat přes počítačovou síť a pro svou činnost využívá běžně dva TCP porty. Přes port 21 jsou posílány příkazy řízení přenosu, na portu 20 jsou pak posílána samotná data. Klient se po připojení k FTP serveru nejdříve autentizuje a poté může na daném serveru manipulovat se soubory, nebo je přenášet z jednoho počítače na druhý.

Jméno skriptu, který monitoruje službu FTP, je ftp–mon.pl Tento monitorovací modul ke své činnosti vyžaduje perlovskou knihovnu `Net::FTP`. Po načtení potřebných dat z databáze a vytvoření RRD souboru se následně skript s pomocí funkcí z výše zmíněné knihovny jednoduše připojí na FTP server, kde se pokusí přihlásit zadanými přihlašovacími údaji. Pokud autentizace dopadne v pořádku, tak si skript nechá vypsat obsah adresáře FTP příkazem `ls`. Čas, který tyto úkony zaberou, je opět získán pomocí časových razítek vytvořených funkcí `clock_gettime`. Naměřená časová odezva je uložena do databázového RRD souboru pro danou adresu.

4.1.9 SSH modul

Secure Shell je protokol, který byl navržen pro zabezpečenou komunikaci mezi dvěma počítači v síti. Tato komunikace umožňuje provádět příkazy na příkazové řádce (tzv. shellu) vzdáleného počítače. SSH byl vytvořen s ohledem na to, aby zabezpečoval autentizaci obou komunikujících subjektů a také šifrování a integritu přenášených dat. Standartně běží SSH server na TCP portu 22. Více je tento protokol rozebrán v dokumentu RFC 4251 [15].

SSH modul je jako všechny ostatní moduly implementován v jazyce Perl a skript je pojmenován ssh–mon.pl. Vzhledem k tomu, že SSH vyžaduje interaktivní přihlášení, tak je

nutno využít speciální knihovnu `Expect`, kterou Perl nabízí. Standartně jsou nejdříve z databáze zjištěna potřebná data. U tohoto modulu je kromě adresy, portu a intervalu nutné znát i přihlašovací jméno a heslo. Po vytvoření databázového souboru následuje monitorovací cyklus, ve kterém je za pomocí knihovny `Expect` proveden SSH příkaz pro připojení na zadaný server. Knihovna `Expect` umožňuje interaktivně reagovat na dotazy SSH serveru. V případě, že je pokus o připojení na SSH server úspěšný, tak je serveru na jeho žádost odesláno příhlášovací heslo. Po přihlášení je pak okamžitě odeslán příkaz `exit` a spojení je ukončeno. Doba, kterou tyto akce zaberou je opět měřena pomocí počátečního a koncového času uložených ve dvou proměnných. Odezva vypočtená odečtením počátečního času od koncového je následně RRD příkazem `update` vložena do RRD databáze.

4.1.10 MySQL modul

MySQL je jeden z nejpoužívanějších databázových systémů, proto je v této aplikaci zařazen modul pro jeho sledování. Mezi výhody MySQL patří zejména jeho multiplatformnost a dobrý výkon. Komunikace s databází probíhá prostřednictvím jazyka SQL. Tento databázový systém většinou běží na TCP portu 3306.

Skript, ve kterém je tento modul implementován má název mysql–mon.pl. Implementace je podobná jako u ostatních modulů, nejdříve se z databáze načtou požadavaná data. Těmi jsou adresa na které běží MySQL systém, port, interval monitorování a u tohoto modulu je také požadováno uživatelské jméno a heslo pro přihlášení do databáze, stejně jako název samotné databáze. Poté je zkontrolována existence RRD databázového souboru a jestliže existuje tak skript přistoupí k vlastnímu monitorování. To se odehrává v nekonečném cyklu v pravidelných časových odstupech daných funkcí `sleep` s parametrem délky intervalu. Modul se pokusí připojit a přihlásit k testované MySQL databázi pomocí funkce `connect`, která je implementována v perlovském modulu `DBD::mysql`. Jestliže tato akce proběhne v pořádku, tak se skript pokusí vypsat všechny tabulky v MySQL databázi a to příkazem `show tables`. Celková odezva, od doby připojení do konce provádění příkazu a odpojení od databáze, je potom klasicky vložena do RRD příkazem `update`.

4.1.11 PostgreSQL modul

Vedle MySQL je PosgreSQL dalším hojně používaným databázovým systémem. Značným kladem PostgreSQL je to, že se jedná o open source software. Stejně jako MySQL pracuje systém s využitím jazyka SQL a PostgreSQL server běží zpravidla na TCP portu 5432.

Soubor, ve kterém je tento modul implementován má název postgresql–mon.pl a jsou v něm, mimo jiné, používány funkce z knihovny `DBD:Pg`. Stejně jako u MySQL modulu je i zde třeba znát kromě monitorované adresy, portu a intervalu také přihlašovací jméno, heslo a název cílové PostgreSQL databáze. V cyklu, ve kterém probíhá monitorování, se k databázi připojujeme opět s využitím funkce `connect` z modulu DBD. Po připojení provedeme nad databází příkaz `SELECT table_name FROM information_schema.tables WHERE table_schema = 'public'`, jehož výsledkem je vypsání tabulek PostgreSQL databáze. Čas provedení těchto operací je stopován stejně jako v předchozích případech. Vypočtená odezva je následně zapsána do odpovídající RRD databáze.

4.1.12 POP3 modul

POP3, tedy Post Office Protocol verze 3, je protokol aplikační vrstvy síťového modelu ISO/OSI. Tento protokol slouží ke stahování e–mailových zpráv z poštovního serveru na klientský počítač. Výchozím portem, na kterém POP3 obvykle pracuje je TCP port 110. Klient komunikuje se serverem využitím krátkých textových zpráv. Nevýhodou služby POP3 je to, že ze serveru stahuje všechny e–mailové zprávy celé, a ne jen jejich hlavičky, jak je tomu u podobného protokolu IMAP. Další informace o protokolu POP3 je možné získat z dokumentu RFC 1725 [10].

Monitorování této služby je implementováno v souboru pop3–mon.pl. Je v něm použita knihovna `Net:Telnet`, s jejíž pomocí bude skript komunikovat s POP3 serverem. Po načtení důležitých údajů z SQL databáze, včetně přihlašovacího jména a hesla, je spuštěn monitorovací cyklus, ve kterém je proveden pokus o spojení se zadaným poštovním serverem. Po vydařeném připojení je zahájená komunikace. Nejdříve je odesláno přihlašovací jméno a následně i heslo. Po každém tomto kroku se čeká na odpověď serveru, která musí mít tvar `+OK`. Po úspěšné autentizaci je na server poslán příkaz `list`, tedy vypsání obsahu dané e–mailové schránky a poté se skript od serveru odpojí. Celá tato procedura je opět

ohraničena počáteční a koncovou časovou značkou, ze kterých je spočtena výsledná odezva, která je potom příkazem `update` zapsána do RRD databázového souboru.

4.1.13 SMTP modul

SMTP, celým názvem Simple Mail Transfer Protocol, je protokol aplikační vrstvy, který slouží posílání e–mailových zpráv mezi tzv. MTA (Mail Transfer Agents). MTA jsou servery z nichž si potom koncový uživatel stahuje své e–maily. SMTP zpravidla funguje na portu TCP číslo 25. Podrobnosti o tomto protokolu jsou uvedeny ve standardu RFC 2821 [7].

Skript provádějící monitorování tohoto protokolu se jmenuje smtp–mon.pl. Pro interaktivní komunikaci se SMTP serverem je znovu využito knihovny `Net:Telnet`. Modul na začátku načte data z SQL databáze, toto monitorování si vystačí s adresou serveru, portem na kterém služba běží a intervalem jednotlivých měření. Jako vždy je také zkontrolována existence databázového RRD souboru pro toto monitorování, což je nutná podmínka pro další běh skriptu. Následuje cyklus, ve kterém se pomocí Telnetu připojíme k SMTP serveru. Jako odpověď po připojení očekáváme kód 220. Následně je na SMTP server odeslána zpráva `EHLO test`, na níž by měl server odpovědět stavovým kódem 250. Pokud vše proběhlo v pořádku, tak dojde k uložení naměřené odezvy do Round Robin databáze.

4.1.14 IMAPs modul

Internet Message Access Protocol [4] je protokol, který podobně jako POP3 umožňuje vzdálený přístup k e–mailové schránce a stahování elektronické pošty na klientský počítač. IMAP umožňuje ale pokročilejší funkce, hlavní výhodou je, že nestahuje všechny informace, ale jen ty, které nezbytně potřebuje a tělo e-mailové zprávy stáhne jen pokud si to uživatel vyžádá. IMAP běžně funguje na portu 143. Jeho zabezpečená verze, kdy komunikace probíhá prostřednictvím SSL, pak většinou běží na TCP portu 993.

Modul je implementován v souboru imaps–mon.pl. Při implementaci je využito knihovny `Net::IMAP::Simple::SSL`, díky čemuž nejsou přihlašovací údaje posílány v otevřené podobě, ale je využito bezpečnostního protokolu SSL (Socket Layer Security). Jakmile se modul přihlásí k serveru tak si nechá vypsat všechny e–mailové schránky daného uživatele a následně se ze serveru odhlásí. Naměřený čas odezvy je uložen do příslušné RRD databáze.

4.1.15 TCP modul

Tento modul umožňuje otestovat dostupnost kteréhokoli TCP portu na cílové adrese, lze tak ověřit zda na něm naslouchá nějaká služba. Modul je implementován ve skriptu tcp–mon.pl. Perlovská část skriptu se jako vždy stará o načtení intervalu s jakým má monitorování probíhat, adresy a cílového portu, stejně jako o vytvoření RRD databáze pro toto monitorování. V monitorovací smyčce je potom volán externí program `nmap`. Aplikaci `nmap` jsou předány parametry adresa, skenovaný port a také parametr `-oG`, který nám poté usnadní práci s výstupem programu. Výstup, který `nmap` vrátí, je poté pomocí tzv. rour poslán do unixových aplikací grep a awk, jimiž je z výstupu získán údaj o tom jaký je stav monitorovaného portu. Je-li skenovaný port označen jako otevřený, tak se monitorování počítá jako úspěšné. Čas odezvy, který je získán z časových značek umístěných před začátkem volání nmapu a po získání jeho výstupu, je uložen do správné RRD databáze, která náleží tomuto sledování. V případě, že je získaný stav portu označen jako filtrovaný, nebo dokonce uzavřený, tak je tento pokus monitorování brán jako neúspěšný a do RRD databáze je vložena tzv. neznámá hodnota.

4.1.16 Upozornění o výpadcích

Každý monitorovací modul si ještě před započetím samotného monitorovacího cyklu načte z databázové tabulky uživatelů informace o tom, kteří uživatelé si přejí zasílat upozornění o výpadcích služby a po jak dlouhé době od výpadku nebo po překročení nastaveného prahu odezvy, má být upozornění zasláno. Poté je vždy v monitorovacím cyklu kontrolováno, zda k výpadku nedošlo a pokud ano, tak se počítá čas jak dlouho výpadek trvá. Pokud naměřená délka výpadku nebo doba odezvy přesáhne hodnotu, kterou si daný uživatel zadal jako mezní pro zaslání upozornění, tak dojde k odeslání tohoto upozornění na zadanou e–mailovou adresu. Upozornění obsahuje informace o tom jaká služba a na které adrese má výpadek, a jak dlouho už je služba nedostupná.

Po odeslání upozornění zároveň dojde k vynulování čítače délky výpadku, což znamená, že další upozornění bude zasláno znovu až po dalším překročení mezní hodnoty zadané uživatelem. Takto je zabráněno tomu, aby v případě delšího výpadku, nebyla uživatelova e–mailová schránka zaplněna upozorněními. Aby odesílání emailu fungovalo správně, tak je třeba mít v systému nainstalovanou aplikaci `ssmtp`.

4.1.17 Bezpečnost monitorování

Vzhledem k tomu, že u mnohých monitorovacích modulů jsou vyžadovány citlivé údaje jako přihlašovací jména a hesla, tak je třeba vzít v úvahu otázku bezpečnosti pro případ, že by se k těmto údajům dostal potencionální útočník. Proto je vhodné pro monitorování těch služeb, u kterých je riziko, že by mohly být nějak zneužity, vytvořit speciální účty pro monitorování, které budou mít omezené pravomoce. Zejména se to týká služby SSH, ale i dalších jako např. služby FTP, kde je zbytečné, aby měl monitorovací účet přístupná všechna data na serveru.

4.2 Webové rozhraní

V následující podkapitole bude popsáno uživatelské webové rozhraní a jeho implementace. Celé webové rozhraní je realizováno pomocí jazyka HTML s využitím skriptovacího jazyka PHP [2].

4.2.1 Přihlášení do systému

Než uživatel začne pracovat s webovým rozhraním, tak je nutné, aby se do něj nejdříve přihlásil. Je to zapotřebí jednak proto, aby nepovolané osoby nezískaly přístup k naměřeným odezvám, byť riziko zneužití těchto informací není velké, ale hlavně proto, že se přes webové rozhraní uskutečňuje nastavení samotného monitorovacího systému a bylo by nežádoucí, aby toto nastavení mohl provádět a měnit kdokoli.

V systému jsou rozlišeny dva typy uživatelů podle akcí, které mohou provádět. Jedná se o administrátorský účet a o účet běžného uživatele. Výchozí administrátorský účet, který je v systému nastaven, má přihlašovací jméno „root“ a heslo k němu je také „root“. K přednastavenému běžnému účtu se lze přihlásit pomocí uživatelského jména „ibp“ a hesla „ibp“. Uživatelské údaje jsou uloženy v SQL databázi, samotné přihlášení potom probíhá v jednoduchém PHP skriptu, který zkontroluje zda jsou přihlašovací data zadaná uživatelem shodná s těmi, která jsou uložena v databázi. Pokud ano, tak uživatel získá přístup do systému. Informace o aktuálním sezení a přihlášeném uživateli jsou uloženy ve speciálních PHP proměnných tzv. sessions. Tento přístup je výhodnější oproti použití cookies, které v dnešní době mnoho uživatelů raději vypíná, a tím pádem se na ně nedá plně spolehnout.

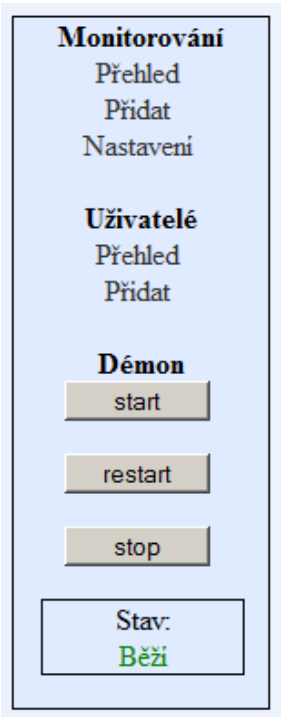
4.2.2 Ovládací panel

Jakmile je uživatel přihlášen, tak může začít pracovat se systémem. Pohyb mezi jeho jednotlivými částmi rozhraní je možný prostřednictvím ovladacího panelu umístěného v levé části stránky. Tento panel je rozdělen na několik podčástí.

V první z nich je informace o aktuálním přihlášeném uživateli a tlačítko, kterým se uživatel může ze systému odhlásit. Pod tímto ovládacím prvkem se nachází nabídka, která umožňuje navigaci po stránce. Obsah této nabídky se liší dle typu přihlášeného uživatele. Běžný uživatel má k dispozici odkazy na přehled všech monitorování, která jsou zadána v systému a také odkaz do nastavení. Nabídka možností pro administrátorský účet je o něco

bohatší. Obsahuje stejné odkazy jako nabídka pro běžného uživatele, krom toho má ale administrátor přístup i do sekce pro přidávání nových monitorování a také do oblasti stránky, ve které má přehled o všech registrovaných uživatelských účtech a odkud může tyto účty spravovat.

Navíc má v ovládacím panelu administrátor zobrazená tlačítka, jejichž použitím může spouštět, restartovat, nebo zastavit monitorovacího démona. V dolní části ovládacího panelu je pro oba typy uživatelů zobrazen prvek indikující stav monitorovacího démona, tedy to, zda démon běží nebo je zastaven. Ovládací panel pro uživatele s administrátorskými právy je zobrazen na obr. 4.1.



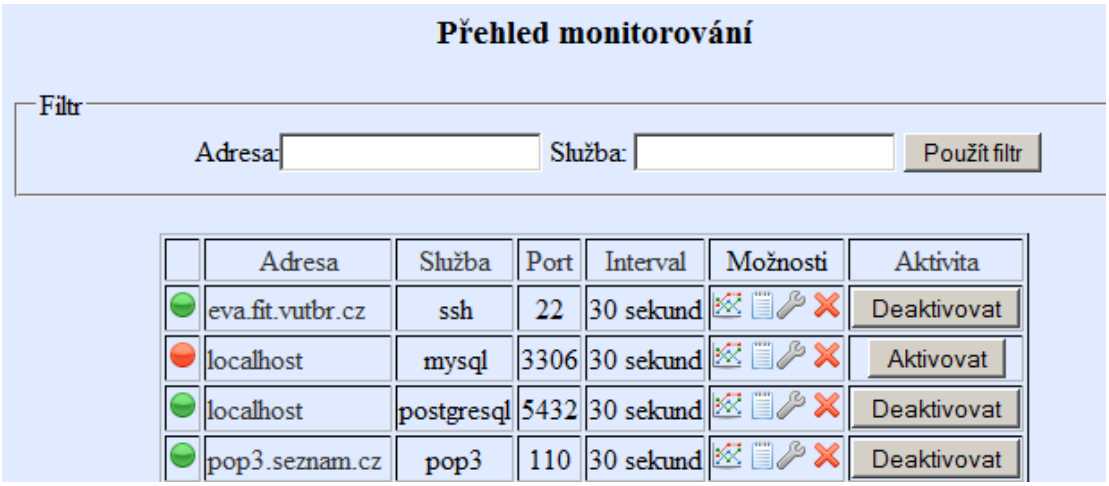
Obrázek 4.1: Hlavní ovládací panel webového rozhraní

4.2.3 Přehled monitorování

V této sekci stránky má uživatel přehled o všech sledováních přítomných v systému. V horní části této stránky se nachází filtr, pomocí něhož si může uživatel vybrat, zda si chce zobrazit sledování jen pro určitou adresy či typ služby. Implicitně jsou zobrazována všechna monitorování. Jejich výpis je v tabulce, jejichž hodnoty jsou načteny z SQL databáze příkazem `SELECT`. Samotný výpis je možné řadit různými způsoby a to kliknutím na název sloupce

v záhlaví tabulky. Takto lze vypisovat hodnoty vzestupně či sestupně podle názvu adresy, služby, portu, délky monitorovacího intervalu či aktivity sledování.

Za zmínku stojí sloupec „Aktivita“, v administrátorském režimu je v něm u každého měření tlačítko, kterým lze dané sledování adresy zastavit nebo znovu spustit (tedy deaktivovat či aktivovat). Dalším důležitým sloupcem je sloupec „Možnosti“, ve kterém jsou k dispozici jednotlivé akce, které se dají s mnitorováním provádět. Běžný uživatel si může pouze zobrazit výsledky pro dané monitorování v grafu či tabulce. Administrátor má navíc možnost upravit nastavení monitorování nebo ho ze systému úplně odstranit.

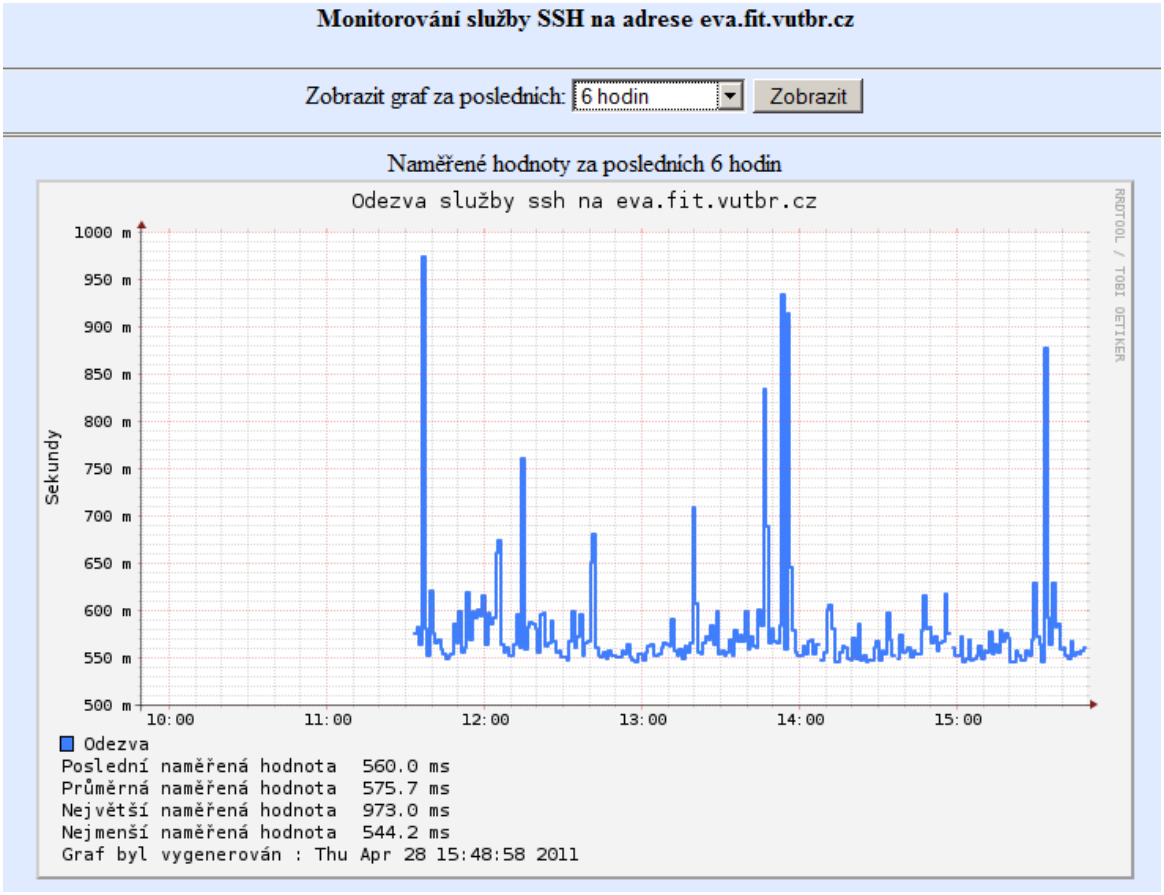


Obrázek 4.2: Přehled monitorovaných adres

4.2.4 Zobrazení grafu

Tento oddíl webového rozhraní je přístupný ze sekce „Přehled monitorování“ po kliknutí na ikonu grafu u zvoleného sledování. Je zde zobrazen graf získaný z naměřených hodnot odezev sledované služby. Implicitně je zobrazován graf hodnot za posledních 6 hodin. Nad grafem je ovládací prvek, kterým může uživatel blíže specifikovat časový rozsah grafu. K dispozici jsou grafy za poslední hodinu, 6 hodin, 12 hodin, den, týden, měsíc a rok. V každém grafu jsou vykresleny odezvy za zvolené období a to tak, že na x–ové ose grafu je zobrazen čas a na y–ové ose je odezva služby naměřená v daném čase. Krom toho jsou ve spodní části grafu vypsány další užitečné údaje jako minimální, maximální a průměrná hodnota pro dané období a také hodnota, která byla naměřená jako poslední. Zároveň každý graf obsahuje

časovou značku, kdy byl vytvořen. Pod grafem je potom odkaz do sekce, kde jsou naměřené hodnoty vypsány do tabulky.



Obrázek 4.3: Zobrazení naměřených odezev v grafu

4.2.5 Zobrazení tabulky

Do této části rozhraní se dá dostat ze sekce „Přehled monitorování“ přes ikonu tabulky u vybraného monitorování nebo přes odkaz pod grafem odezev. V tabulce je možnost zobrazit naměřené výsledky maximálně za poslední den. Nad tabulkou je tlačítko, kterým lze kontrolovat, zda mají být zobrazeny všechny naměřené hodnoty nebo pouze výpadky služby. Také se zde nachází odkaz zpět na graf daného monitorování.

4.2.6 Úpravy monitorování

Do této sekce má přístup pouze administrátor a to přes odkaz v sekci „Přehled monitorování“. Může zde upravit parametry zvoleného monitorování. Měnit lze hodnoty adresy serveru, zkratku služby, portu, intervalu monitorování a u služeb, které to vyžadují také přihlašovací jméno, heslo nebo také název monitorované databáze.

Nutno podotknout, že pokud se uživatel rozhodne změnit adresu, název služby či interval, tak to bude mít za následek ztrátu dosud naměřených dat. Je to z toho důvodu, že při změně těchto parametrů je nutné vytvořit zcela nový databázový RRD soubor. Jakmile uživatel potvrdí změny monitorování, tak PHP skript odešle příkaz RESTART na monitorovací server, který znovu spustí sledování s novými parametry.

4.2.7 Přidat monitorování

Sekce přístupná odkazem z hlavního ovladacího panelu. Přidávat nová monitorování může pouze administrátor. Do kolonky v horní části uživatel zadá adresu serveru a v tabulce potom zaškrtne všechny služby, které chce na adrese sledovat a u každé služby nastaví potřebné údaje. Lze nastavit monitorování služeb HTTP, ICMP (Ping), DNS, FTP, SSH, MySQL, PostgreSQL, POP3, SMTP, IMAPs a skenování TCP portu.

Zvláštní položkou je Jiná služba. Takto může uživatel zadat monitorování nějaké jiné vlastní služby, která není implicitně zabudována v systému. Nutnou podmínkou je, aby se v adresáři „scripts“ monitorovacího démona nacházel příslušný skript, pro monitorování této služby. Zároveň musí být splněno, že název (zkratka) služby, kterou uživatel vyplní do tohoto formuláře, musí být obsažena také v názvu zmíněného monitorovacího skriptu a to tak, aby bylo dodrženo pravidlo, podle kterého jsou vytvářeny názvy monitorovacích skriptů, tak jak bylo popsáno v kapitole o monitorovacích modulech.

Dalším podstatným parametrem je interval monitorování. K dispozici jsou intervaly 15 a 30 sekund, dále také 2 minuty, 5 minut nebo 10 minut. Implicitně je nastavena hodnota 30 sekund, ale uživatel sám musí zhodnotit, který interval je pro něj nejvhodnější. Platí, že čím kratší interval, tím je měření přešnější. Je to ovšem vykoupeno větší velikostí RRD databázového souboru a také zvýšeným zatížením síťového provozu. Po stisknutí tlačítka Přidat nové monitorování dojde k odeslání vyplněného formuláře. PHP skript na straně ser-

veru zkontroluje, zda byly všechny údaje zadány správně a pokud ano, tak je monitorování úspěšně vloženo do SQL databáze. V opačném případě je uživatel upozorněn na chybu.

4.2.8 Nastavení a upozornění na výpadky

V oddíle „Nastavení“ má uživatel k dispozici formulář, kde si může změnit své přihlašovací heslo. Také si zde může nastavit e–mailovou adresu, na kterou mu budou zásílána upozornění v případě, že dojde k výpadku některé ze sledovaných služeb. V sekci „Upozornění“ pak vidí všechna nastavená upozornění a zároveň může přidávat nová upozornění nebo upravovat ta stávající. Posílání e–mailových zpráv je uskutečněno PHP funkcí `mail`, proto musí být webový server správně nastaven, aby tato funkce pracovala správně.

Administrátor má navíc v sekci „Nastavení“ možnost nastavit adresu a port, na kterém má probíhat komunikace s monitorovacím démonem. Implicitní hodnoty jsou localhost a port 10001.

4.2.9 Přehled a přidávání uživatelů

Tyto dvě sekce webového rozhraní jsou přístupné pouze uživateli s administrátorským oprávněním. V sekci „Přehled uživatelů“ je zobrazena tabulka, ve které jsou vypsáni všichni uživatelé a jejich údaje. Administrátorovi je umožněno vybraného uživatele odebrat z databáze pomocí ikony „Odstranit“. V sekci „Přidat uživatele“ se nachází formulář, který slouží k přidání nového uživatele do systému. U něj je, krom ostatních údajů, nutné zadat i e–mailovou adresu. Na tuto adresu bude uživateli zasláno jeho přihlašovací heslo, které je náhodně vygenerováno PHP skriptem. K poslání této zprávy je opět využita PHP funkce `mail`. Dále je možno zatrhnout, zda má mít uživatel administrátorské pravomoce či ne.

5 Závěr

Tato bakalářská práce měla jako hlavní cíl provedení návrhu a implementace monitorovacího systému, který bude schopen sledovat široké spektrum služeb a který bude umožňovat jednoduché zobrazení naměřených hodnot. Čtenář byl také seznámen s oblastí monitorování počítačových sítí a sledování odezev služeb a aplikací v těchto sítích. Rozlišeny byly dvě základní myšlenky, jež se objevují v problematice monitorování. První z nich je sledování založené na aktivním přístupu, kdy je určitým způsobem napodobováno chování skutečného uživatele. Při této metodě je do sítě generován uměle vytvořený provoz a lze tak sledovat jakým způsobem na něj síť reaguje a odhalit možné problémy. Druhý přístup, který byl v této práci představen, je založen na pasivním sledování datových toků proudících v síti. Narozdíl od prvního způsobu je u pasivního monitorování sledovaný provoz vytvořen skutečnými uživatelskými akcemi. Analýza naměřených dat dá správci sítě možnost sledovat výkonnost sítě při běhu různých typů služeb. Výhody a nevýhody obou metod, tedy aktivního i pasivního sledování, byly rozebrány s výsledným zhodnocením, že robustní monitorovací nástroj by měl vhodně propojit oba způsoby monitorování.

V další kapitole byl poté proveden návrh monitorovací aplikace, která funguje na principu aktivního monitorování. Navržený systém byl následně implementován. Aplikace se skládá ze dvou částí, a to z webového rozhraní a z monitorovacího démona. Monitorovací démon byl implementován v jazyce Perl, který obsahuje bohatou sadu knihoven, jež usnadnily práci při tvorbě jednotlivých monitorovacích modulů. Pro ukládání naměřených dat byla zvolena aplikace RRDTool. Stejná aplikace je potom použita při vytváření grafů. Druhou částí aplikace je její webové rozhraní. To slouží k jednoduchému ovládání systému a také k zobrazení naměřených odezev.

Během testování byly sledovány odezvy jednotlivých služeb na různých serverech a naměřené hodnoty odpovídaly aktuální situaci v síti a jejímu zatížení v době měření.

Implementovaný monitorovací nástroj nabízí spoustu prostoru k další práci a rozšiřování. Zejména stojí za úvahu, jak do něj vhodně zakomponovat modul, který by sloužil k pasivnímu měření provozu. Nicméně i stávající moduly aktivního monitorování by bylo možno rozšířit o další možnosti a provádět sofistikovanější sledování daných služeb, které by lépe napodobilo činnost skutečných uživatelů služby. Vzhledem k tomu, že je využit nástroj RRDTool, který nabízí mnoho možností jakým způsobem vykreslovat výsledné grafy, tak by bylo možné obohatit webové rozhraní o nastavení, kterými by si uživatel mohl přizpůsobit zobrazování grafů, tak jak mu nejlépe vyhovuje.

Literatura

[1] Perl programming documentation.
URL <http://perldoc.perl.org/index.html>

[2] Achour, M.; Betz, F.; Dovgal, A.; aj.: PHP Manual.
URL <http://php.net/manual/en/index.php>

[3] Case, J.; Fedor, M.; Schoffstall, M.; aj.: Simple Network Management Protocol (SNMP). RFC 1157 (Historic), Květen 1990.
URL <http://www.ietf.org/rfc/rfc1157.txt>

[4] Crispin, M.: INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501 (Proposed Standard), Březen 2003.
URL <http://www.ietf.org/rfc/rfc3501.txt>

[5] Eisenzopf, J.: Unix Daemons in Perl. Leden 2000.
URL <http://www.webreference.com/perl/tutorial/9/index.html>

[6] Fielding, R.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Historic), 1999.
URL <http://www.ietf.org/rfc/rfc2616.txt>

[7] Klensin, J.: Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), Duben 2001.
URL <http://www.ietf.org/rfc/rfc2821.txt>

[8] Mockapetris, P.: Domain names - concepts and facilities. RFC 1034 (Standard), Listopad 1987.
URL <http://www.ietf.org/rfc/rfc1034.txt>

[9] Mockapetris, P.: Domain names - implementation and specification. RFC 1035 (Standard), Listopad 1987.
URL <http://www.ietf.org/rfc/rfc1035.txt>

[10] Myers, J.; Rose, M.: Post Office Protocol - Version 3. RFC 1725 (Standard), Listopad 1994.
URL <http://www.ietf.org/rfc/rfc1725.txt>

[11] Oetiker, T.: RRDTool Documentation. 2009.
URL <http://oss.oetiker.ch/rrdtool/doc/index.en.html>

[12] Postel, J.: Internet Control Message Protocol. RFC 792 (Standard), Září 1981.
URL <http://www.ietf.org/rfc/rfc792.txt>

[13] Postel, J.; Reynolds, J.: File Transfer Protocol. RFC 959 (Standard), Říjen 1985.
URL <http://www.ietf.org/rfc/rfc959.txt>

[14] Stevens, W. R.; Fenner, B.; Rudoff, A. M.: *UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API*. Addison-Wesley, Listopad 2003, iSBN: 0-13-141155-1.

[15] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), Leden 2006.
URL <http://www.ietf.org/rfc/rfc4251.txt>

A Obsah DVD

Přiložené DVD obsahuje:

- Všechny zdrojové soubory monitorovacího systému
- Virtuální disk pro aplikaci VirtualBox, na kterém je celý monitorovací systém ve spustitelné podobě
- Instalační program pro aplikaci VirtualBox
- Soubor README, ve kterém je popsáno spuštění monitorovacího systému ve VirtualBoxu